EARTHQUAKE ENGINEERING AND STRUCTURAL DYNAMICS
Earthquake Engng. Struct. Dyn. **27**, 641–648 (1998)

CONTROL OF STRUCTURAL SEISMIC RESPONSE BY SELF-RECURRENT NEURAL NETWORK (SRNN)

YU-AO HE*,† AND JIANJUN WU‡
Department of Civil Engineering, Tianjin University, Tianjin, 300072, China

SUMMARY

A new paradigm called self-recurrent neural network (SRNN) is proposed. Two SRNNs are utilized in a control system, one as an emulator and the other as a controller. To guarantee convergence and for faster learning, an approach using adaptive learning rate is developed by Lyapunov function. Finally, the neural network control algorithm is developed for on-line control of structural seismic response in real time. Simulation-results have shown that it can effectively control structural seismic response and make it consist with the desired response. © 1998 John Wiley & Sons, Ltd.

KEY WORDS: self-recurrent neural network; learning rate; structural seismic response; control in real time

INTRODUCTION

With increase research activities in the field of structural control in recent years, a number of structural control methods have been proposed. Some of the widely used structural control methods are:[1–4] optimal control, pole assignment, independent modal space control, instantaneous optimal control, polynomial control, pulse control, hybrid control, etc. Most of these control algorithms require the analysis and/or identification of the system. Although some structural models can be routinely developed, there are many other aspects of the problem that are more difficult to identify and incorporate in control algorithms. Structures do not behave exactly as their mathematical models do. Inherent delay in the control loop is another aspect of the control problem that is difficult to compensate for. In short, there are many sources of non-linearity, uncertainty, and measurement noise, which result in poor mathematical models and less-effective control algorithms: so much so that there seems to be a need for AI engineering theory to develop more effective control algorithms for the active control of structures.

In searching for more effective control algorithms, the Artificial Neural Network (ANN) is emerging as a promising tool and has shown great potential for the purposes of control,[5] It has many attributes, such as massive parallelism, adaptability, robustness, and the inherent capability to handle non-linear systems. At present, several neural network control methods have been proposed to solve the above problems.[6,7] Most used the feedforward neural network, and the backpropagation training algorithm to solve the dynamical problems; however, the feedforward network is a static mapping and without the aid of tapped delays it does not represent a dynamic system mapping; on the other hand, its slow convergence cannot accommodate to the stochastic earthquake. For all these reasons, a shorter training time for the neural network model, called Self-Recurrent Neural Network (SRNN), is developed. The SRNN model is a dynamic mapping in a way the Fully Connected Recurrent Neural Network (FRNN) is, but there are no interlinks among neurons in the

* Correspondence to: Yu-ao He, Department of Civil Engineering, Tianjin University, Tianjin, 300072, China
† Professor
‡ Ph.D. Student

Contract/grant sponsor: National Science Foundation of China

CCC 0098–8847/98/070641–08$17·50

Received 4 June 1997

hidden layer so that it has considerably fewer weights than the FRNN and the network is simplified considerably. In this paper, the convergence of the SRNN-based system is investigated, and an analytical method based on the Lyapunov function to find the adaptive learning rates for the SRNN is proposed. The neural network control algorithm is developed for on-line control of structural seismic response.

SELF-RECURRENT NEURAL NETWORK AND DYNAMIC BACKPROPAGATION ALGORITHM

Self-recurrent neural network structure

A three-layer self-recurrent neural network architecture is shown in Figure 1. The mathematical model for the SRNN is

input layer:
$$I_i(t) = X_i(t) \tag{1}$$

hidden layer:
$$\begin{gathered} H_j(t) = f[\mathrm{net}_j(t)] \\ \mathrm{net}_j(t) = W_j^{(2)} H_j(t-1) + \sum_i W_{ij}^{(1)} X_i(t) \end{gathered} \tag{2}$$

output layer:
$$O(t) = \sum_j W_j^{(3)} \cdot H_j(t) \tag{3}$$

where for each discrete time t, $X_i(t)$ is the ith input to the SRNN, $\mathrm{net}_j(t)$ is the sum of inputs to the jth recurrent neuron, $H_j(t)$ is the output of the jth recurrent neuron, and $O(t)$ is the output of the SRNN. Here $f[*]$ is the sigmoid function, and $W_{ij}^{(1)}, W_j^{(2)}$ and $W_j^{(3)}$ are input, recurrent, and output weight vectors, respectively.

Dynamic backpropagation algorithm for SRNNs

For neuro-emulator, let $y(t)$ and $\hat{y}(t) = O(t)$ be the structural response and output of the SRNN emulator, then an error function for training cycles can be defined as

$$E_{\mathrm{M}} = \tfrac{1}{2}[y(t) - \hat{y}(t)]^2 \tag{4}$$

The gradient of error in equation (4) with respect to output, recurrent and input weights, respectively, are given by

$$\frac{\partial E_{\mathrm{M}}}{\partial W_j^{(3)}} = -e_{\mathrm{m}}(t)\frac{\partial \hat{y}(t)}{\partial W_j^{(3)}} = -e_{\mathrm{m}}(t)\frac{\partial O(t)}{\partial W_j^{(3)}} = -e_{\mathrm{m}}(t)\cdot H_j(t) \tag{5}$$

$$\frac{\partial E_{\mathrm{M}}}{\partial W_j^{(2)}} = -e_{\mathrm{m}}(t)\frac{\partial O(t)}{\partial W_j^{(2)}} = -e_{\mathrm{m}}(t)\frac{\partial O(t)}{\partial H_j(t)}\frac{\partial H_j(t)}{\partial W_j^{(2)}} = -e_{\mathrm{m}}(t)\, W_j^{(3)} \delta_j^{\mathrm{M}}(t) \tag{6}$$

$$\frac{\partial E_{\mathrm{M}}}{\partial W_{ij}^{(1)}} = -e_{\mathrm{m}}(t)\frac{\partial O(t)}{\partial W_{ij}^{(1)}} = -e_{\mathrm{m}}(t)\frac{\partial O(t)}{\partial H_j(t)}\frac{\partial H_j(t)}{\partial W_{ij}^{(1)}} = -e_{\mathrm{m}}(t)\, W_j^{(3)} \beta_{ij}^{\mathrm{M}}(t) \tag{7}$$

$$\delta_j^{\mathrm{M}}(t) = f'[\mathrm{net}_j(t)][H_j(t-1) + W_j^{(2)}\delta_j^{\mathrm{M}}(t-1)] \tag{8}$$

$$\beta_j^{\mathrm{M}}(t) = f'[\mathrm{net}_j(t)][X_j(t) + W_j^{(2)}\beta_{ij}^{\mathrm{M}}(t-1)] \tag{9}$$

$e_m(t) = y(t) - \hat{y}(t)$ is the error between the structure and neuro-emulator response. Suppose initial condition $\delta_j^{\mathrm{M}}(0) = 0$ and $\beta_{ij}^{\mathrm{M}}(0) = 0$, $\delta_j^{\mathrm{M}}(t)$ and $\beta_{ij}^{\mathrm{M}}(t)$ can be calculated.

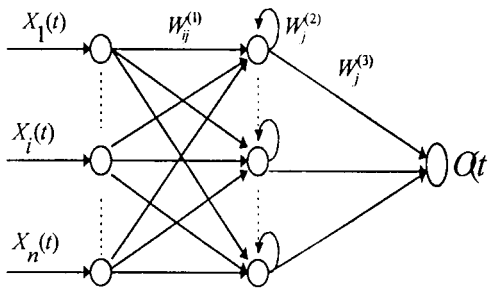

Figure 1. Self-recurrent neural network architecture

The weights can be adjusted following a gradient method, i.e., the update rule of the weights becomes

$$W(t+1) = W(t) + \eta(t)\left(-\frac{\partial E_{\mathrm{M}}}{\partial W}\right) + \alpha \Delta W \tag{10}$$

where W represents $W_{ij}^{(1)}$, $W_j^{(2)}$ and $W_j^{(3)}$, $\eta(t)$ is called an adaptive learning rate, α is a momentum factor. From equations (5)–(9), which lead to $\partial E_{\mathrm{M}}/\partial W$.

$X_i(t) = [\,y(t-1), \ldots, y(t-n), u(t), u(t-1), \ldots, u(t-m), \ddot{x}_{\mathrm{g}}(t), \ddot{x}_{\mathrm{g}}(t-1), \ldots, \ddot{x}_{\mathrm{g}}(t-k)]$. $u(t)$ is control command and $\ddot{x}_{\mathrm{g}}(t)$ is earthquake ground acceleration. By backpropagating the error function E_{M} to adjust the weights, the neuro-emulator can be trained to reach a desired accuracy for modelling the dynamic behaviour of the real structure and send its output which represents structural actual response to train neuro-controller.

For neuro-controller, suppose SRNN control law is

$$u(t) = g[\,y(t-1), \ldots, y(t-n), u(t-1), \ldots, u(t-m), \ddot{x}_{\mathrm{g}}(t), \ddot{x}_{\mathrm{g}}(t-1), \ldots, \ddot{x}_{\mathrm{g}}(t-k), y_{\mathrm{r}}(t)] \tag{11}$$

where $y_{\mathrm{r}}(t)$ is the desired control response of structure which is excited by earthquake.

The error function (4) is also modified by the SRNN emulator by replacing $y(t)$ and $\hat{y}(t)$ with $y_{\mathrm{r}}(t)$ and $y(t)$, respectively, i.e.,

$$E_{\mathrm{c}} = \tfrac{1}{2}\,[y_{\mathrm{r}}(t) - y(t)]^2 \tag{12}$$

Like neuro-emulator, the weights of neuro-controller can be adjusted as

$$\frac{\partial E_{\mathrm{c}}}{\partial V_j^{(3)}} = -\,e_{\mathrm{c}}(t)\frac{\partial y(t)}{\partial u(t)}\frac{\partial u(t)}{\partial V_j^{(3)}} = -\,e_{\mathrm{c}}(t) Q_j(t)\frac{\partial y(t)}{\partial u(t)} \tag{13}$$

$$\frac{\partial E_{\mathrm{c}}}{\partial V_j^{(2)}} = -\,e_{\mathrm{c}}(t)\frac{\partial y(t)}{\partial u(t)}\frac{\partial u(t)}{\partial Q_j(t)}\frac{\partial Q_j(t)}{\partial V_j^{(2)}} = -\,e_{\mathrm{c}}(t)\, V_j^{(3)}\delta_j^{\mathrm{c}}(t)\frac{\partial y(t)}{\partial u(t)} \tag{14}$$

$$\frac{\partial E_{\mathrm{c}}}{\partial V_{ij}^{(1)}} = -\sum_j e_{\mathrm{c}}(t)\frac{\partial y\,(t)}{\partial u(t)}\frac{\partial u(t)}{\partial Q_j(t)}\frac{\partial Q_j(t)}{\partial V_{ij}^{(1)}} = -\sum_j e_{\mathrm{c}}(t)\, V_j^{(3)}\beta_{ij}^{\mathrm{c}}(t)\frac{\partial y(t)}{\partial u(t)} \tag{15}$$

$$\delta_j^{(\mathrm{c})}(t) = \frac{\partial Q_j(t)}{\partial V_j^{(2)}} = G'[\mathrm{net}_j(t)]\,[Q_j(t-1) + V_j^{(2)}\delta_j^{\mathrm{c}}(t-1)] \tag{16}$$

$$\beta_{ij}^{(\mathrm{c})}(t) = \frac{\partial Q_j(t)}{\partial V_{ij}^{(1)}} = G'[\mathrm{net}_j(t)]\,[I_i(t) + V_j^{(2)}\beta_{ij}^{\mathrm{c}}(t-1)] \tag{17}$$

where for each discrete time t, $I_i(t)$ is the ith input to the SRNN, $Q_j(t)$ is the output of the jth recurrent neuron, and $U(t)$ is the output of the SRNN. Here $g[*]$ is the sigmoid function, and $V_{ij}^{(1)}$, $V_j^{(2)}$ and $V_j^{(3)}$ are input, recurrent and output weight vectors, respectively.

Similar, $\delta_j^c(t)$ and $\beta_{ij}^c(t)$ can be calculated as neuro-emulator. $e_c(t) = y_r(t) - \hat{y}(t)$ is the error between the desired and actual response of the structure. The factor $\partial y(t)/\partial u(t)$ represents the sensitivity of the structure with respect to control force. Since structural property is normally unknown, the sensitivity term $\partial y(t)/\partial u(t)$ is unknown. This unknown value can be estimated by using the neuro-emulator. When the neuro-emulator is in training, the dynamic behaviour of the neuro-emulator is close to the unknown structure, i.e., $\hat{y}(t) \to y(t)$, where $\hat{y}(t)$ is the output of the neuro-emulator. Once the training process is done, we assume the sensitivity can be approximated as

$$\frac{\partial y(t)}{\partial u(t)} \approx \frac{\partial \hat{y}(t)}{\partial u(t)} \tag{18}$$

Note that $\hat{y}(t) = O(t)$ and

$$\frac{\partial y(t)}{\partial u(t)} = \frac{\partial \hat{y}(t)}{\partial u(t)} = \frac{\partial O(t)}{\partial u(t)} = \sum_j W_j^{(3)} \cdot \frac{\partial H_j(t)}{\partial u(t)} \tag{19}$$

$$\frac{\partial H_j(t)}{\partial u(t)} = f'[\mathrm{net}_j(t)] \frac{\partial \mathrm{net}_j(t)}{\partial u(t)} \tag{20}$$

Since inputs to the neuro-emulator is $[y(t-1), \ldots, y(t-n), \quad u(t), \quad u(t-1), \ldots, u(t-m), \quad \ddot{x}_g(t), \ddot{x}_g(t-1), \ldots, \ddot{x}_g(t-k)]$

Thus

$$\mathrm{net}_j(t) = W_j^{(2)} H_j(t-1) + \sum_{i=1}^{n} W_{ij}^{(1)} y(t-i) + \sum_{i=n_1}^{n_1+m} W_{ij}^{(1)} u(t-i+n_1) + \sum_{i=n_2}^{n_2+k} W_{ij}^{(1)} \ddot{x}_g(t-i+n_2)$$

$$\frac{\partial \mathrm{net}_j(t)}{\partial u(t)} = W_{n+1,j}^{(1)} \quad n_1 = n+1, \; n_2 = n+m+2 \tag{21}$$

$$\frac{\partial y(t)}{\partial u(t)} = \sum_j W_j^{(3)} \cdot f'[\mathrm{net}_j(t)] \cdot W_{n+1,j}^{(1)} \tag{22}$$

The weights of the SRNN controller can be determined as follows:

$$V(t+1) = V(t) + \eta(t) \left(-\frac{\partial E_c}{\partial V} \right) + \alpha \Delta V(t) \tag{23}$$

where V represents $V_{ij}^{(1)}$, $V_j^{(2)}$, $V_j^{(3)}$, $\eta(t)$ is called an adaptive learning rate, α is a momentum factor. Equations (13)–(17) and (22) lead to $\partial E_c/\partial V$.

Convergence and stability

The stability of the SRNN control system is determined by the convergence of the SRNNs backpropagation algorithm which is determined by the value of the learning rate. For faster convergence, a guideline in selecting the learning rate property, which leads to adaptive learning rate, is developed.

A discrete-type Lyapunov function can be given by

$$L(t) = \tfrac{1}{2} e^2(t) \tag{24}$$

where $e(t)$ represents the error in the learning process, $e(t) = e_m(t)$ or $e_c(t)$.

Thus, the change of the Lyapunov function due to the training process is obtained by

$$\Delta L(t) = \tfrac{1}{2} [e^2(t+1) - e^2(t)] \tag{25}$$

The error difference due to the learning can be represented by

$$e(t+1) = e(t) + \frac{\partial e(t)}{\partial W}\Delta W \tag{26}$$

where ΔW represents a change in an arbitrary weight vector.

For neuro-emulator, from the update rule of equations (5)–(10)

$$\Delta W = -\eta(t)\frac{\partial E_{\mathrm{M}}}{\partial W} = \eta(t)e_{\mathrm{m}}\frac{\partial e_{\mathrm{m}}}{\partial W} = \eta(t)e_{\mathrm{m}}\frac{\partial O}{\partial W} \tag{27}$$

$$\Delta L(t) = \Delta e_{\mathrm{m}}(t)[e_{\mathrm{m}}(t) + \tfrac{1}{2}\Delta e_{\mathrm{m}}(t)] = -\lambda e_{\mathrm{m}}^2(t) \tag{28}$$

$$\lambda = \frac{1}{2}\left\|\frac{\partial O(t)}{\partial W}\right\|^2 \eta(t)(2-\eta(t)\left\|\frac{\partial O(t)}{\partial W}\right\|^2 \tag{29}$$

Let

$$g(t) = \frac{\partial O(t)}{\partial W}, \quad g_{\max} = \max\|g(t)\| \quad \text{and} \quad \eta_1 = \eta g_{\max}^2{}^{8}$$

Then

$$\lambda = \frac{1}{2}\|g(t)\|^2\eta(t)\left(2 - \frac{\eta_1\|g(t)\|^2}{g_{\max}^2}\right) \geqslant \frac{1}{2}\|g(t)\|^2\eta(t)(2-\eta_1) > 0 \tag{30}$$

Then, the convergence of the SRNN emulator will be guaranteed if $\eta(t)(2-\eta_1) > 0$ or $\eta_1(2-\eta_1)/g_{\max}^2 > 0$, i.e., $0 < \eta(t) < 2/g_{\max}^2$. However, the maximum learning rate which guarantees the most rapid or optimal convergence correspond to $\eta_1 = 1$, i.e.,

$$\eta(t) = \frac{1}{g_{\max}^2} \tag{31}$$

For neuro-controller, from the update rule of equations (13)–(23)

$$\Delta V = -\eta(t)e_{\mathrm{c}}\frac{\partial e_{\mathrm{c}}}{\partial V} = \eta(t)e_{\mathrm{c}}\frac{\partial y(t)}{\partial u(t)}\frac{\partial u(t)}{\partial V} \tag{32}$$

$$\Delta L(t) = -\eta(t)e_{\mathrm{c}}^2(t)\left(\frac{\partial y(t)}{\partial u(t)}\right)^2\left\|\frac{\partial u(t)}{\partial V}\right\|^2 + \tfrac{1}{2}\eta^2(t)e_{\mathrm{c}}^2(t)\left(\frac{\partial y(t)}{\partial u(t)}\right)^4\left\|\frac{\partial u(t)}{\partial W}\right\|^4 = -\lambda e_{\mathrm{c}}^2(t) \tag{33}$$

Comparing equation (33) with equation (29), it can be seen that both conditions are similar, except that the sensitivity $\partial y(t)/\partial u(t)$ needs to be incorporated in the SRNN controller.

Since, from equation (22)

$$\frac{\partial y(t)}{\partial u(t)} = \sum_j W_j^{(3)} f'[\mathrm{net}_j(t)]\, W_{n+1,j}^{(1)}$$

where $0 < f'[\cdot] < 0.5$. Define

$$W_{n+1\max}^{(1)} = \max_t \|W_{n+1}^{(1)}(t)\|, \quad \|W_{n+1}^{(1)}(t)\| = \max_j |W_{n+1,j}^{(1)}|$$

then

$$\frac{\partial \hat{y}(t)}{\partial u(t)} \leqslant h\|W^{(3)}(t)\|\,\|f'[\mathrm{net}_j(t)]\,W_{n+1}^{(1)}(t)\| \leqslant \frac{h}{2}W_{\max}^{(3)}(t)\,W_{n+1\max}^{(1)}(t) = S_{\max} \tag{34}$$

where h is the number of recurrent neurons in the hidden layer. It is similar to the neuro-emulator, from equations (33) and (34). The SRNN controller will converge if any $\eta(t)$ is chosen as $0 < \eta(t) < 2/S_{\max}^2 g_{\max}^2$ and the optimal convergence rate is

$$\eta(t) = \frac{1}{S_{\max}^2 g_{\max}^2} \tag{35}$$

where

$$g_{\max} = \max_t \|g(t)\|, \quad g(t) = \partial u(t)/\partial V$$

CONTROL ALGORITHM BASED ON SRNN

The block diagram of the SRNN-based control system is shown in Figure 2. An unknown structure is identified by a system identifier, called the SRNN emulator. The inputs to the SRNN emulator are the previous structural response $y(k-1)$, the control signal $u(k)$, and the earthquake ground acceleration $\ddot{x}_g(k)$, and the output of the SRNN emulator is the predicted response. An error function for a training cycling for SRNN emulator is defined as $E_m = \frac{1}{2}[y(k) - \hat{y}(k)]^2$. The error between the output $\hat{y}(k) = O(k)$ of the SRNN emulator and structural response is computed for each iteration, and is used to adjust the weights of the SRNN emulator. The neuro-emulator can be trained to reach a desired accuracy for modelling the dynamic behaviour of the real structure.

After the neuro-emulator is obtained, the neuro-controller can be trained. The SRNN emulator provides information about the structure to the neuro-controller. The SRNN controller is used to drive the unknown structure so that the error between structural response and desired response is minimized. An error function for a training cycling for SRNN emulator is defined as $E_c = \frac{1}{2}[y_r(k) - y(k)]^2$. The inputs to the SRNN controller are the desired response $y_r(k)$, the previous structural response $y(t-1)$, the earthquake ground acceleration $\ddot{x}_g(k)$, and the previous control signal $u(k-1)$, and the output of the SRNN controller is the control signal $u(k)$ to the structure. By using the method from the previous section the weights of the SRNN controller are readjusted.

SIMULATION RESULTS

To demonstrate the performance of the SRNN emulator and controller in the control of structure under dynamic loading, a two-storey building model is chosen as the structure to be controlled. Assume the properties of the building are as follows: $m_1 = 60\,\text{kg}$, $m_2 = 50\,\text{kg}$, $k_1 = 1000\,\text{N/m}$, $k_2 = 833{\cdot}3\,\text{N/m}$,

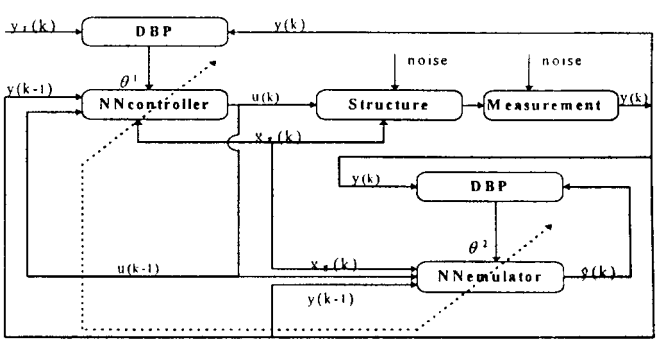

Figure 2. The block diagram of the SRNN-based control system

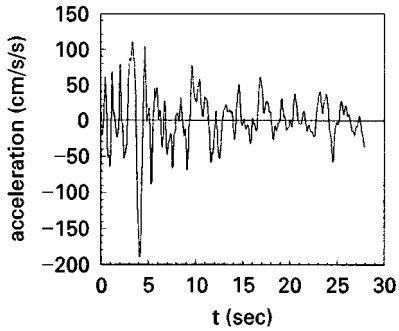

Figure 3. The earthquake ground acceleration

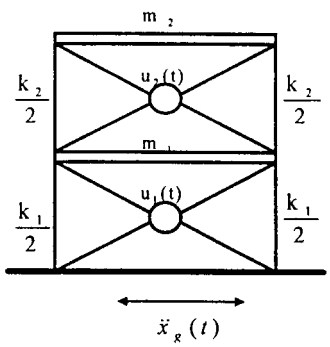

Figure 4. The controlled structure

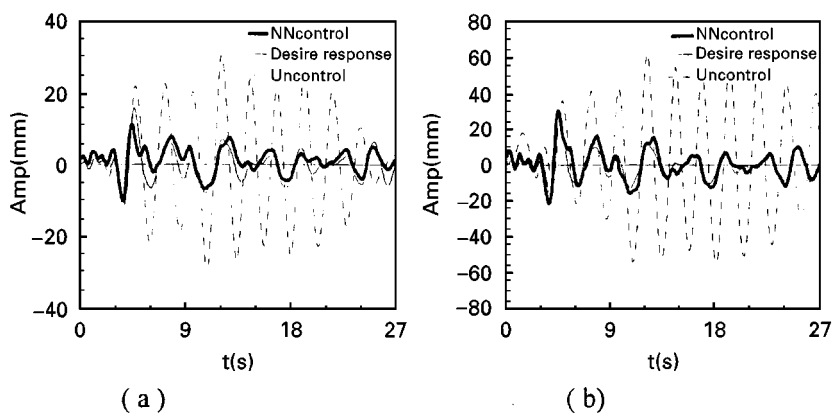

Figure 5. Structural response before and after control: (a) the first storey and; (b) the second storey

$c_1 = c_2 = 0$. An earthquake ground acceleration is shown in Figure 3. Figure 4 is the controlled structure.

The SRNN emulator trained here is constructed with a four-node input layer, a nine-node hidden layer, and a one-node output layer. The input data of the four nodes in the input layer are the two consecutive displacements, the acceleration on the ground floor, and the control command. The output of the

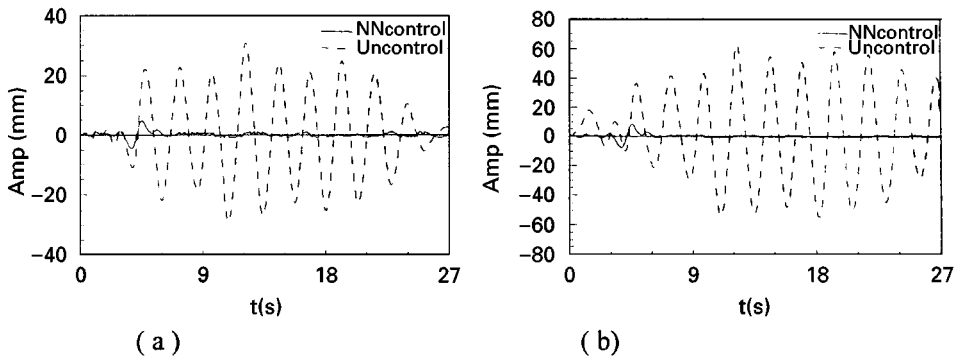

Figure 6. Structural response before and after control: (a) the first storey and; (b) the second storey

neuro-emulator is the displacement of the first or second storey at the next time step. The SRNN controller is modelled as a three-layer and a five-node input layer, eleven-node hidden layer, and a one-node output layer. The input data of the five nodes in neuro-controller are the two consecutive displacements, the acceleration on the ground floor, the previous control command, and the desired control response. The output of the neuro-controller is the next control command, representing the force needed to be generated to the building for control purpose. The first 250 points taken from the 1350 points record of the earthquake is the data set used for training of the SRNN control system which includes the neuro-emulator and the neuro-controller. Adaptive learning rates were used starting from the initial rate of $\eta = 0{\cdot}1$.

The desired response in Figure 5 are calculated by the instantaneous optimal control algorithm. In Figure 6, the target response are chosen to zero. From them, we have achieved a great promising effect. And in training process, all of the four training processes do not take more than 45 cycles, a cycle being 250 epochs.

CONCLUSION

A neural paradigm called Self-Recurrent Neural Network (SRNN) is presented here. A generalized dynamic back-propagation algorithm (DBP) is developed. To guarantee convergence and for faster learning, an approach using adaptive learning rates is developed by introducing a Lyapunov function. The SRNN-based control system is tested for its on-line control of structural seismic response. Results from computer-simulation studies have shown: (1) the SRNN algorithm-based control system needs neither the mathematical model of the structure nor a control method, it only needs the desired control response of the structure as its control target; (2) the SRNN has many attributes, such as massive parallelism, adaptability, robustness, and the inherent capability to handle non-linear system.

REFERENCES

1. J. N. Yang, Z. Li and S. Vongchavalitkul, 'A generalization of optimal control theory: linear and nonlinear control', *J. Engng. Mech.*, ASCE **120**, 266–283 (1994).
2. J. N. Yang and S. C. Liu, 'Stable controller for instantaneous optimal control', *J. Engng. Mech.*, ASCE **118**, 1612–1630 (1992).
3. J. N. Yang, A. K. Agrawal and S. Chen, 'Optimal polynomial control for seismically excited non-linear and hysteretic structures', *Earthquake Engng. and Struct. Dyn.* **25**, 1211–1230 (1996).
4. T. Kobori, N. Koshika, K. Yamada and Y. Ikeda, 'Seismic response controlled structure with active mass drive system. Part I: Design, Part II: Verification, *Earthquake Engng. and Struct. Dyn.* **20**, 1211–1230 (1991).
5. K. S. Narendra, *et al.* 'Identification and control of dynamic systems using neural networks', *IEEE Trans. Neural network* **1**, 4–27 (1990).
6. J. Ghaboussi and A. Joghataie, 'Active control of structures using neural networks', *J. Engng. Mech. ASCE* **121**(5), 555–567 (1995).
7. H. M. Chen, K. H. Tsai, G. Z. Qi and J. C. S. Yang, 'Neural network for structure control', *J. Comput. Civil Engng.*, **9**(2), 168–175 (1995).
8. M. M. Polycarpou and P. A. Ioannou, 'Learning and convergence analysis of neural-type structural networks', *IEEE Trans. Neural Network* **3**(1), 39–50 (1992).

